**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

Курсовая РАБОТА

**по дисциплине «Программирование»**

Тема: Хранение и обработка числовых данных на основе файлов и массивов

|  |  |  |
| --- | --- | --- |
| Студент гр. 4354 |  | Чучалин И. В. |
| Преподаватель |  | Калмычков В.А. |

Санкт-Петербург

2024

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Студент Чучалин И.В. | | |
| Группа 4354 | | |
| Тема работы: хранение и обработка числовых данных на основе файлов и массивов. | | |
| Исходные данные: реализовать алгоритм на основе разбиения программы на набор функций с параметрами. Программа не должна содержать стандартных и библиотечных типов и связанных с ними функций. Должны быть учтены все составляющие формулировки задания и выполнены указанные действия, причем в качестве результата должны быть предложены все варианты, удовлетворяющие заданию. Программа должна предоставлять возможность неоднократного выполнения действий. | | |
| Содержание пояснительной записки:  Исходная формулировка задания, математическая постановка задачи, описание алгоритма, текст программы, вывод программы, выводы. | | |
| Предполагаемый объем пояснительной записки:  Не менее 15 страниц. | | |
| Дата выдачи задания: 22.11.2024 | | |
| Дата сдачи реферата: | | |
| Дата защиты реферата: | | |
| Студент |  | Чучалин И.В. |
| Преподаватель |  | Калмычков В.А. |

**Аннотация**

В данной курсовой работе рассмотрена работа с числовыми данными, файлами, массивами и функциями на примере языка C++. Необходимо корректно составить математическую постановку задачи, обработать файл с исходными данными, при помощи двухмерных или трёхмерных массивов организовать хранение данных и с помощью функций реализовать программу для решения геометрической задачи. Результатом выполнения курсовой работы служит программа, способная решать геометрическую задачу с разными начальными условиями и реагирующая на некорректные данные.

**Summary**

In this course work, work with numeric data, files, arrays and functions is considered using the example of the C++ language. It is necessary to correctly formulate a mathematical statement of the problem, process the file with the initial data, organize data storage using two-dimensional or three-dimensional arrays and use functions to implement a program to solve a geometric problem. The result of the course work is a program capable of solving a geometric problem with different initial conditions and responding to incorrect data.

Содержание:

[Исходная формулировка задания 5](#_Toc185597506)

[Ограничения на исходные данные 5](#_Toc185597507)

[Математическая постановка задачи 5](#_Toc185597508)

[Дано 5](#_Toc185597509)

[Найти 5](#_Toc185597510)

[Способ решения 5](#_Toc185597511)

[Вычислим длины сторон: 5](#_Toc185597512)

[Контрольный пример 6](#_Toc185597513)

[Организация интерфейса пользователя 6](#_Toc185597514)

[Макет ввода/вывода 6](#_Toc185597515)

[Реализация вывода в консоль 7](#_Toc185597516)

[Внутренний формат хранения данных 7](#_Toc185597517)

[Работа с файлами 7](#_Toc185597518)

[Описание функций 8](#_Toc185597519)

[Синтаксис 8](#_Toc185597520)

[Назначение 8](#_Toc185597521)

[Описание алгоритма 9](#_Toc185597522)

[Текст программы 9](#_Toc185597523)

[Вывод программы 13](#_Toc185597524)

[Выводы 15](#_Toc185597525)

# Исходная формулировка задания

Дано N точек на плоскости. Для всех треугольников, образуемых любыми тремя точками, определить, является ли треугольник остроугольным.

# Ограничения на исходные данные

Будем обрабатывать не больше 50 точек ().

# Математическая постановка задачи

Дано точек на плоскости.

Найтивсе остроугольные треугольники, образуемые любыми тремя точками.

Способ решения

Вычислим длины сторон:







Вычислим внутренние углы по теореме косинусов:













Получив внутренние углы, можно определить остроугольность треугольника – если все углы меньше 90 градусов, то треугольнк остроугольный.

Контрольный пример

Рассмотрим треугольник с вершинами (872;-98), (78;7), (78;3):

A math equations on a white background

Description automatically generated

A math equations on a white background

Description automatically generated

# Организация интерфейса пользователя

## Макет ввода/вывода

|  |  |
| --- | --- |
| Fresult1. | formed triples of coordinates: |
| Fresult2. | d\_\_d. d\_\_d.d\_\_d d\_\_d.d\_\_d d\_\_d.d\_\_d d\_\_d.d\_\_d d\_\_d.d\_\_d d\_\_d.d\_\_d |
| Fresult3. | Task: Find all acute-angled triangles formed by any three points |
| Fresult4. | number of acute-angled triangles:d\_\_d |
| Fresult5. | acute-angled triangles: |
| Fresult6. | d\_\_d (d\_\_d). 1. (d\_\_d;d\_\_d) 2. (d\_\_d;d\_\_d) 3. (d\_\_d;d\_\_d) |
| Fresult7. | this file is not open or does not exist. please, restart the programm |
| Fprotocol1. | result file is not open or does not exist. please, restart the programm |
| Fprotocol2. | this file is not open or does not exist. please, restart the programm |
| Fprotocol3. | Specified number of points (d\_\_d) has been reduced to N |
| Fprotocol4. | Readed number of points: d\_\_d |
| Fprotocol5. | Real number of points: d\_\_d |
| Fprotocol6. | number of acute-angled triangles: d\_\_d |
| Fprotocol7. | end of inputFile |
| Fprotocol8. | there is no paired coordinate for x (d\_\_d.d\_\_d) |
| Fprotocol9. | d\_\_d. d\_\_d.d\_\_d d\_\_d.d\_\_d d\_\_d.d\_\_d d\_\_d.d\_\_d d\_\_d.d\_\_d d\_\_d.d\_\_d |
| Fprotocol10. | This is an acute-angled triangle. |
| Fprotocol11. | This is not an acute-angled triangle. |
| Fprotocol12 | Selected file: |
| Cin1. | d |
| Cin2. | \n |
| Cout1. | protocol file is not open or does not exist. please, restart the programm |
| Cout2. | result file is not open or does not exist. please, restart the programm |
| Cout3. | available files: |
| Cout4. | d. “ind.txt” |
| Cout5. | select the file to open : |
| Cout6. | error: this file is not exist. please, press enter to try again: |
| Cout7. | this file is not open or does not exist. please, restart the programm |
| inpurFile1. | s\_s |
| inputFile2. | d\_\_d.d\_\_d |

## Реализация вывода в консоль

|  |  |
| --- | --- |
| Библиотека | Вывод |
| iostream | std::cout << |

## Внутренний формат хранения данных

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| N | const unsigned | Ограничение колличества считываемых точек |
| fresult | fstream | Файл результата |
| fprotocol | fstream | Файл протокола |
| P[] | bool | Масиив для хранения истиности остроугольности |
| inputFile | fstream | Входной файл |
| x[], y[] | float | Массив считываемых точек |
| XP[][], YP[][] | long double | Двумерный массив с тройками координат |
| tryAgain | char | Вспомогательная |
| file | unsigned |
| fileselected | bool |
| big\_number | int | Необработанное считанное число точек |
| real\_number | Обработанное считанное число точек |
| number | Реальное число точек |
| result | Число острых треугольников |
| index | Число троек координат |
| i, j , k | Перемнные для счета |
| s | char | Вспомогательная |
| tmp\_x | float |
| tmp\_y |
| longest | int |
| x, y ,z |
| x1, y1, x2, y2 | double |

# Работа с файлами

Для работы с файлами используется библиотека «fstream».

|  |  |
| --- | --- |
| Метод | Описание |
| fstream f | Создание файловой переменной |
| f.open(“file.txt”, ios::out) | Открытие файла на запись |
| f.open(“file.txt”, ios::in) | Открытие файла на чтение |
| filein.seekg(-1, ios::cur); | Перемещение курсора назад относительно текущего положения |
| filein >> noskipws | Чтение пробелов, символов перехода на новую строку |
| f.is\_open() | Определяет, открыт ли файл |
| f.eof() | Проверка на конец файла |
| f << | Запись данных в файл |
| f >> | Чтение данных из файла |

# Описание функций

## Синтаксис

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя | Тип возвращаемого значения | Параметры | | |
| Тип | Имя | Изменение |
| readFile | int | Fsream | inputFile | & |
| Float | x[N], y[N] | - |
| int | big\_number, real\_number, number | & |
| pointsCreation | void | Float | x[N], y[N] | - |
| Long double | XP[100000][3], YP[100000][3] | - |
| Int | real\_number | - |
| Int | index | & |
| proccess | int | Long double | XP[100000][3], YP[100000][3] | - |
| Int | result | & |
| Int | index | - |
| distance | double | double | x1, y1, x2, y2 | - |
| main | void | - | - | - |

## Назначение

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Имя | Возращ. знач. | Внешние изменения | Параметры | | | | Назначение |
| Входной | Выходной | Модифицируемый | Транзитный |
| readFile | 0, -1, -2, -3, number | - | inputFile | x[], y[], big\_number, real\_number, number | fprotocol | - | Считывает координаты из файла, отфильтровывая все лишнее |
| pointsCreation | - | - | x[], y[] | XP[][], YP[][], index | fprotocol, fresult | real\_number | Создает уникальные тройки координат |
| process | 0 | - | XP[][], YP[][], index | result | fprotocol | index | Проверяет треугольник на остроугольность |
| distance | sqrt(pow(x1 - x2, 2) + pow(y1 - y2, 2)) | - | x1, y1, x2, y2 | - | - | x1, y1, x2, y2 | Высчитывает длины сторон треугольника |
| main | - | Открытие файлов, резервация памяти для массивов | - | - | fprotocol, fresult | - | Основная функция программы |

# Описание алгоритма

Программа берет точки и задает треугольники, затем проверяет их на остроугольность. Результат работы программы выводится в файл «result.txt». Все действия программы фиксируются в протоколе «protocol.txt». Блок-схема работы программы представлена ниже:

# Текст программы

**#include <iostream>;**

**#include <iomanip>;**

**#include <math.h>;**

**#include <fstream>;**

**using namespace std;**

**const unsigned N = 50;**

**fstream fresult, fprotocol;**

**bool P[10000];**

**int readFile(fstream& inputFile, float x[N], float y[N], int& big\_number, int& real\_number, int& number);**

**void pointsCreation(float x[N], float y[N], long double XP[100000][3], long double YP[100000][3], int real\_number, int& index);**

**int process(long double XP[10000][3], long double YP[10000][3], int& result, int index);**

**double distance(double x1, double y1, double x2, double y2);**

**void main(void)**

**{**

**fstream inputFile;**

**float x[N], y[N]; long double XP[10000][3], YP[10000][3]; char tryAgain = '!'; unsigned file; bool fileSelected = false;**

**int big\_number = 0, real\_number = 0, number = 0, result = 0, index = 0, i, j = 0;**

**fresult.open("result.txt", ios::out);**

**fprotocol.open("protocol.txt", ios::out);**

**if (!fprotocol.is\_open())**

**{**

**cout << "protocol file is not open or does not exist. please, restart the programm";**

**return;**

**}**

**if (!fresult.is\_open())**

**{**

**cout << "result file is not open or does not exist. please, restart the programm";**

**fprotocol << "result file is not open or does not exist. please, restart the programm";**

**return;**

**}**

**//file selection**

**while (fileSelected != true)**

**{**

**cout << skipws << "available files:\n";**

**cout << "\t0. \"in.txt\"\n";**

**cout << "\t1. \"in1.txt\"\n";**

**cout << "\t2. \"in2.txt\"\n";**

**cout << "\t3. \"in3.txt\"\n";**

**cout << "\t4. \"in4.txt\"\n";**

**cout << "\t5. \"in5.txt\"\n";**

**cout<<"select the file to open : ";**

**cin >> file;**

**switch (file)**

**{**

**case 0:**

**{**

**inputFile.open("in.txt", ios::in);**

**fileSelected = true;**

**fprotocol << "selected file: in.txt\n";**

**break;**

**}**

**case 1:**

**{**

**inputFile.open("in1.txt", ios::in);**

**fileSelected = true;**

**fprotocol << "selected file: in1.txt\n";**

**break;**

**}**

**case 2:**

**{**

**inputFile.open("in2.txt", ios::in);**

**fileSelected = true;**

**fprotocol << "selected file: in2.txt\n";**

**break;**

**}**

**case 3:**

**{**

**inputFile.open("in3.txt", ios::in);**

**fileSelected = true;**

**fprotocol << "selected file: in3.txt\n";**

**break;**

**}**

**case 4:**

**{**

**inputFile.open("in4.txt", ios::in);**

**fileSelected = true;**

**fprotocol << "selected file: in4.txt\n";**

**break;**

**}**

**case 5:**

**{**

**inputFile.open("in5.txt", ios::in);**

**fileSelected = true;**

**fprotocol << "selected file: in5.txt\n";**

**break;**

**}**

**default:**

**{**

**cout << "\nerror: this file is not exist. please, press enter to try again: " << noskipws;**

**cin.ignore();**

**cin.get();**

**system("cls");**

**fileSelected = false;**

**break;**

**}**

**}**

**}**

**//checking file opening**

**if (!inputFile.is\_open())**

**{**

**cout << "this file is not open or does not exist. please, restart the programm";**

**fprotocol << "this file is not open or does not exist. please, restart the programm";**

**fresult << "this file is not open or does not exist. please, restart the programm";**

**return;**

**}**

**//checking number of points**

**switch (readFile(inputFile, x, y, big\_number, real\_number, number))**

**{**

**case 0:**

**{**

**cout << "since the number of points is zero, programm cannot be executed";**

**fprotocol << "since the number of points is zero, programm cannot be executed";**

**return;**

**}**

**case -1:**

**{**

**cout << "the number of points cannot be less than zero";**

**fprotocol << "the number of points cannot be less than zero";**

**return;**

**}**

**case -2:**

**{**

**cout << "file is empty";**

**fprotocol << "file is empty";**

**return;**

**}**

**case -3:**

**{**

**cout << "file reading error";**

**fprotocol << "file reading error";**

**return;**

**}**

**}**

**if (big\_number != 0)**

**{**

**//cout << "specified number of points (" << big\_number << ") has been reduced to " << N << '\n';**

**fprotocol << "specified number of points (" << big\_number << ") has been reduced to " << N << '\n';**

**}**

**//cout << "readed number of points: " << number << '\n';**

**//cout << "real number of points: " << real\_number << '\n';**

**fprotocol << "readed number of points: " << number << '\n';**

**fprotocol << "real number of points: " << real\_number << '\n';**

**//process**

**pointsCreation(x, y, XP, YP, real\_number, index);**

**process(XP, YP, result, index);**

**//cout << "number of acute-angled triangles:" << result;**

**fprotocol << "number of acute-angled triangles:" << result;**

**fresult << "Task: Find all acute-angled triangles formed by any three points\n";**

**fresult << "number of acute-angled triangles:" << result << '\n';**

**fresult << "acute-angled triangles:\n";**

**for (i = 0; i < index; i++)**

**{**

**if (P[i] == true)**

**{**

**j++;**

**fresult << setw(2) << left << j << " (" << setw(3) << i << ")." << right << "\t1." << '(' << setw(7) << left << XP[i][0] << ";" << setw(7) << right << YP[i][0] << ')' << "\t2." << '(' << setw(7) << left << XP[i][1] << ";" << setw(7) << right << YP[i][1] << ')' << "\t3." << '(' << setw(7) << left << XP[i][2] << ";" << setw(7) << right << YP[i][2] << ')' << endl;**

**}**

**}**

**inputFile.close();**

**}**

**int readFile(fstream& inputFile, float x[N], float y[N], int& big\_number, int& real\_number, int& number)**

**{**

**int i = 0; char s = '!'; float tmp\_x, tmp\_y;**

**inputFile >> number;**

**//checking number of points**

**if (inputFile.eof())**

**{**

**return -2;**

**}**

**else if (number < 0)**

**{**

**return -1;**

**}**

**else if (number == 0)**

**{**

**return 0;**

**}**

**//programm will work**

**else**

**{**

**if (number > N)**

**{**

**big\_number = number;**

**number = N;**

**}**

**while(i < number)**

**{**

**inputFile << skipws;**

**inputFile >> tmp\_x;**

**if (inputFile.eof())**

**{**

**fprotocol << "end of inputFile\n";**

**return number;**

**}**

**else**

**{**

**inputFile << noskipws;**

**do inputFile >> s;**

**while (s == ' ' || s == '\t');**

**if (s == '\n')**

**{**

**//реакция на то, что есть только х**

**fprotocol << "there is no paired coordinate for x (" << tmp\_x << ")\n";**

**continue;**

**}**

**else if (inputFile.eof())**

**{**

**//реакция на то, что есть только х**

**fprotocol << "there is no paired coordinate for x (" << tmp\_x << ")\n";**

**return number;;**

**}**

**else**

**{**

**inputFile.seekg(-1, ios::cur);**

**s = '!';**

**inputFile << skipws;**

**inputFile >> tmp\_y;**

**if (inputFile.eof())**

**{**

**//реакция на конец файла**

**fprotocol << "end of inputFile\n";**

**return number;**

**}**

**else**

**{**

**x[i] = tmp\_x;**

**y[i] = tmp\_y;**

**//cout << i+1 << ".\t" << x[i] << ' ' << y[i] << '\n';**

**fprotocol << i + 1 << ".\t" << x[i] << ' ' << y[i] << '\n';**

**i++;**

**real\_number = i;**

**inputFile << noskipws;**

**while (!inputFile.eof() && s != '\n') inputFile >> s;**

**s = '!';**

**}**

**}**

**}**

**}**

**return number;**

**}**

**return -3;**

**}**

**void pointsCreation(float x[N], float y[N], long double XP[100000][3], long double YP[100000][3], int real\_number, int& index)**

**{**

**int i, j, k;**

**for (i = 0; i < real\_number; i++)**

**{**

**for (j = i + 1; j < real\_number; j++)**

**{**

**for (k = j + 1; k < real\_number; k++)**

**{**

**XP[index][0] = x[i];**

**XP[index][1] = x[j];**

**XP[index][2] = x[k];**

**YP[index][0] = y[i];**

**YP[index][1] = y[j];**

**YP[index][2] = y[k];**

**index++;**

**}**

**}**

**}**

**//cout << endl;**

**fresult << "formed triples of coordinates:\n";**

**for (i = 0; i < index; i++) {**

**//cout << i+1 << ".\t" << XP[i][0] << " " << YP[i][0] << "\t\t" << XP[i][1] << " " << YP[i][1] << "\t\t" << XP[i][2] << " " << YP[i][2] << endl;**

**fresult << i + 1 << ".\t" << XP[i][0] << " " << YP[i][0] << "\t\t" << XP[i][1] << " " << YP[i][1] << "\t\t" << XP[i][2] << " " << YP[i][2] << endl;**

**fprotocol << i + 1 << ".\t" << XP[i][0] << " " << YP[i][0] << "\t\t" << XP[i][1] << " " << YP[i][1] << "\t\t" << XP[i][2] << " " << YP[i][2] << endl;**

**}**

**return;**

**}**

**int process(long double XP[10000][3], long double YP[10000][3], int& result, int index)**

**{**

**int longest, x, y, z, i;**

**for (i = 0; i < index; i++)**

**{**

**x = distance(XP[i][0], YP[i][0], XP[i][1], YP[i][1]);**

**y = distance(XP[i][1], YP[i][1], XP[i][2], YP[i][2]);**

**z = distance(XP[i][2], YP[i][2], XP[i][0], YP[i][0]);**

**longest = z;**

**if (longest < x) {**

**z = longest;**

**longest = x;**

**x = z;**

**}**

**if (longest < y) {**

**z = longest;**

**longest = y;**

**y = z;**

**}**

**if (x \* x + y \* y > longest \* longest) {**

**//cout << i + 1 << ".\t" << "This is an acute-angled triangle.\n";**

**fprotocol << i + 1 << ".\t" << "This is an acute-angled triangle.\n";**

**P[i] = true;**

**result++;**

**}**

**else**

**{**

**//cout << i + 1 << ".\t" << "This is not an acute-angled triangle.\n";**

**fprotocol << i + 1 << ".\t" << "This is not an acute-angled triangle.\n";**

**P[i] = false;**

**}**

**}**

**return 0;**

**}**

**double distance(double x1, double y1, double x2, double y2)**

**{**

**return sqrt(pow(x1 - x2, 2) + pow(y1 - y2, 2));**

**}**

# Вывод программы

Контрольный пример: входной файл (рис. 5), файл с результатом (рис. 6), протокол (рис. 7).

A screenshot of a computer

Description automatically generated

Рисунок 5.

A screen shot of a computer program

Description automatically generated

Рисунок 6.

A screenshot of a computer

Description automatically generated

...

A screen shot of a computer screen

Description automatically generated

...



Рисунок 7.

# Выводы

В ходе данной курсовой работы были рассмотрены следующие понятия: обработка и хранение числовых данных, файловые переменные, режимы открытия и закрытия файла, проверка на то, открыт ли файл, считывание и запись информации в файл, динамическая память, указатели, арифметика указателей, удаление данных, одномерные, двумерные и трёхмерные массивы, функции, параметры функций, возвращаемое значение функций, ссылки, типы данных, условные конструкции, циклы и функция main. Также на практике была реализована программа, решающая геометрическую задачу на пересечение фигур на координатной плоскости.